Ludwig-Maximilians-Universität München
Institut für Informatik
PD Dr. Matthias Schubert
Dr. Andreas Züfle, Dr. Erich Schubert

München, 29.4.2014

Algorithmen und Datenstrukturen

SS 2014

Übungsblatt 3: O-Notation II

Besprechung: 8.5.–13.5.

Abgabe aller mit **Punkten** versehenen Aufgaben bis 7.5.

Aufgabe 3-1 O-Notation II — 5 Punkte

Geben Sie an und begründen Sie kurz, ob die folgenden Aussagen wahr oder falsch sind.

(a) $3n^2 + 30n + 300 = \Omega(n^2)$

(b) $3n^2 + 30n + 300 = \Theta(n^2)$

(c) $\sqrt{n+1} = O(\sqrt{n})$

(d) $3^{n+1} = O(3^n)$

(e) $x^n = O(n)$

Aufgabe 3-2 O-Notation III — 4 Punkte

Geben Sie an, für welche $x \in \mathbb{R}^+$ die folgende Gleichung gilt:

$$\sum_{i=0}^{n} x^i = O(n)$$

Beweisen Sie Ihr Ergebnis.

Aufgabe 3-3 O-Notation (Fibonaccizahlen)

Die Fibonaccifolge ist definiert als: $f_0 = 1$, $f_1 = 1$, $f_n = f_{n-1} + f_{n-2}$ (für $n > 1$)

(a) Geben Sie einen rekursiven Algorithmus zur Berechnung der n-ten Fibonaccizahl in Java-Code an und schätzen Sie dessen Laufzeit mittels O-Notation ab.

(b) Überlegen Sie, wie dieser Algorithmus effizienter umgesetzt werden kann. Geben Sie auch für diese Umsetzung eine Laufzeitabschätzung in O-Notation an.

Aufgabe 3-4 Maximum-Subarray

Das Maximum-Subarray Problem besteht darin, in einem Array aus ganzen Zahlen der Länge n die **nicht leere** Teilfolge von Zahlen zu finden, deren Summe maximal ist.

Beispiel: Für die Eingabefolge $[x_0, ..., x_9]$

$$[12, -32, 38, 22, -27, 32, 17, -13, -10, 10]$$

ist die maximale Summe einer Teilfolge 82 für die Teilfolge $[x_2, ..., x_6]$.

Ein naiver Algorithmus zur Lösung dieses Problems ist der folgende: Berechne für jedes mögliche Start- und Endelement die dazwischenliegende Summe und merke Dir die größte Summe und die entsprechenden Start- und Endelemente. Als Java-Algorithmus ausgedrückt:

```
public void naiveSolution(Integer[] x) {
  int maxSum = 0;
  for (int u = 0; u < x.length; u++) {
    for (int o = 0; o < x.length; o++) {
      // bestimme die Summe der Elemente der Teilfolge von [x_u, ..., x_o]
      int sum = 0;
      for (int i = u; i <= o; i++) {
        sum = sum + x[i];
      }
      // vergleiche die gefundene Summe mit der bis jetzt maximalen
      maxSum = Math.max(maxSum, sum);
    }
  }
}
```

(a) Schätzen Sie die Zeitkomplexität des obigen Algorithmus ab und begründen Sie Ihre Angabe kurz.

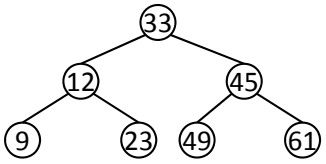
(b) Geben Sie die Idee für einen Algorithmus an, der obiges Problem in linearer Komplexität löst und demonstrieren Sie diese am obigen Zahlenbeispiel.

(c) Implementieren Sie Ihre Lösungsidee aus Aufgabe (b) in Java. Das Programm soll dabei die Summe, das erste und das letzte Element der maximalen Teilfolge als Ergebnis ausgeben. Die Eingabe des Inputarrays als Konstante ist ausreichend.

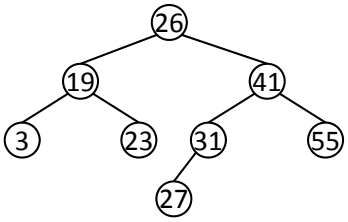
Aufgabe 3-5 AVL-Bäume I **3 Punkte**

Gegeben seien die folgenden drei binären Bäume. Entscheiden Sie welcher der Bäume ein AVL-Baum ist und welcher nicht. Begründen Sie Ihre Aussagen.

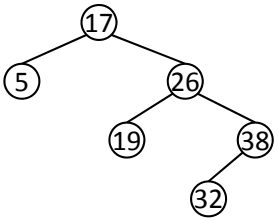
(a)



(b)



(c)



Aufgabe 3-6 Einfügen in AVL-Bäumen

Fügen Sie in einen anfangs leeren AVL-Baum nacheinander die Monatsnamen **entsprechend ihrer Reihenfolge im Jahresverlauf** ein. Gehen Sie beim Vergleich der Schlüssel von einer **lexikographischen Ordnung** aus. Falls es beim Einfügen zu Rebalancierungen kommt, zeichnen Sie bitte den AVL-Baum davor und danach und sagen Sie, ob es sich um eine einfache oder doppelte Rotation handelt. Zeichnen Sie auch den endgültigen AVL-Baum. (zu verwendende Monatskürzel: **Jan, Feb, Mar, Apr, Mai, Jun, Jul, Aug, Sep, Okt, Nov, Dez**)